МИНИМИЗАЦИЯ ЧИСЛА СРАВНЕНИЙ В ХУДШЕМ СЛУЧАЕ В АЛГОРИТМЕ ПОИСКА ПОРЯДКОВЫХ СТАТИСТИК

С.М. Авдошин,
профессор, руководитель отделения программной инженерии факультета бизнес-информатики Государственного университета — Высшей школы экономики, e-mail: savdoshin@hse.ru.
М.П. Шатилов,
студент четвертого курса отделения программной инженерии факультета бизнес информатики Государственного университета — Высшей школы экономики, e-mail: mshatilov@hse.ru.
Адрес: 105187, Москва, ул. Кирпичная, 33/5, отделение программной инженерии.

В работе предложена постановка задачи параметрической оптимизации алгоритма выбора М. Блума, Р. Флойда, В. Пратта, Р. Райвеста и Р. Тайрьяна с линейным временем работы в наихудшем случае. Определяется значение параметра, обеспечивающего минимальную теоретическую верхнюю границу числа сравнений в худшем случае для выполнения алгоритма. Исследуется полученная в результате численных экспериментов зависимость числа сравнений в алгоритме поиска порядковых статистик от различных значений параметра.

Ключевые слова: бинарные сравнения, теория рекурсии, оптимизация алгоритмов, алгоритм с линейным временем работы, алгоритм генерации перестановок.

Введение

Понятие порядковых статистик связано с задачей поиска в линейно упорядоченном множестве из n элементов того элемента, который будет стоять на месте t, если расположить элементы по убыванию. Такой элемент называется t-ой порядковой статистикой. Максимальный элемент – это первая порядковая статистика, а минимальный элемент – n-ая порядковая статистика. Медианой называется порядковая статистика с номером $n/2$.

На математическом семинаре в 1929-1930 г. Г. Штейнгауз сформулировал задачу нахождения минимального числа теннисных матчей, требуемых для определения первого и второго игроков в турнире, если имеется $n > 2$ игроков. Он утверждал, что решением этой задачи является $n - 2 + \lceil \log_2 n \rceil$. Однако, доказательство этого факта оказалось ошибочным [1].

В 1969 г. А. Адьян и М. Собель вывели формулу, дающую хорошую верхнюю оценку [2]:

$$V_t(n) \le n - t + (t-1) \cdot \lceil \log_2 (n + 2 - t) \rceil, \quad n \ge t \qquad (1)$$

где $V_t(n)$ – количество сравнений, необходимое для нахождения t-го элемента в порядке убывания из n элементов.

В 1971 г. М. Блум открыл метод, требующий в худшем случае только $O(n \cdot \log\log n)$ сравнений. Подход М. Блума к этой задаче дал толчок к развитию нового класса методов, которые привели к построению алгоритма с линейной сложностью в худшем случае, принадлежащему Р. Райвесту и Р. Тайрьяну [3]. В русскоязычной литературе алгоритм носит название выбор с линейным временем работы в наихудшем случае М. Блума, Р. Флойда, В. Пратта, Р. Райвеста и Р. Тайрьяна. Для краткости далее будем называть его алгоритмом выбора.

Приведем описание данного алгоритма в классическом труде Д. Кнута [4].

Шаг 1. Разобьем элементы на $2q+1$ групп по 7 элементов в каждой и отсортируем каждую группу.

Шаг 2. Найдем медиану x из $2q+1$ медиан, полученных на шаге 1.

Шаг 3. Множество из $n-1$ элементов, отличных от x, разбиваем на три подмножества (*рис. 1*):

- $4q+3$ элементов, о которых известно, что они больше x (область B);
- $4q+3$ элементов, о которых известно, что они меньше x (область C);
- $6q$ элементов, отношение которых к x неизвестно (области A, D).

Выполнив дополнительно $4q$ сравнений, можно в точности сказать, какие элементы из областей A и D меньше x.

Шаг 4. Пусть мы нашли r элементов, больших x, и $n-r-1$ элементов, меньших x. Если $t=r+1$, то x и будет ответом; если $t<r+1$, то нужно найти t-й элемент в порядке убывания из r больших элементов; и если $t>r+1$, то нужно найти $(t-r-1)$-й элемент в порядке убывания из $n-r-1$ меньших элементов.

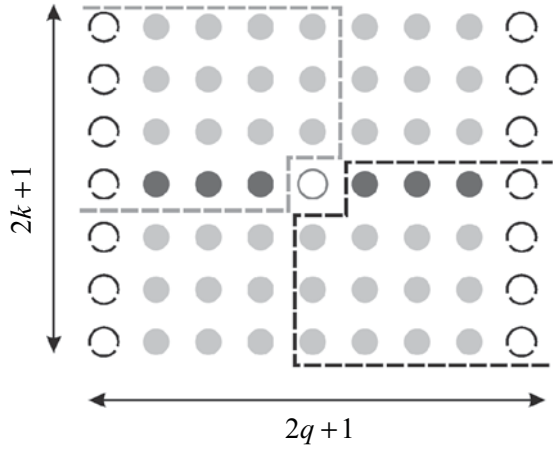

Рис 1. Входной массив после выполнения шага 3 алгоритма при $k=3$

Заметим, что в [5] и [6] предлагается разбиение на группы по 5 элементов, в [4] на 7 элементов.

Возникает вопрос, почему в описанном алгоритме выбрано количество элементов в группах, равное 5 или 7?

Параметризация алгоритма выбора

Параметризируем алгоритм переменной k, через которую определим количество элементов $2k+1$ в каждой из $2q+1$ групп, на которые разбивается массив A длины $n=h-l+1$. Алгоритм выбора, параметризированный константой k представлен на *рис. 2.* Этот алгоритм ищет *t*-ю порядковую статистику в диапазоне от l до h массива A.

```
Tth_Element(l, h, t)
1  const k = ...     // k ∈ {2,3,4,5}
2   n = h − l + 1
3   if (n < c) then
4      Insertion_Sort(l, h)
5      return A[t]
6   else
7      q = (n / (2k + 1) − 1) / 2
8      for i ← 0 to 2q do MK(l + i, 2q + 1, 2k + 1)
9      m ← Tth_Element(l + k · (2q + 1), l +
       + (k + 1) · (2q + 1) − 1, l + (2q + 1) · k + q)
10     Partition_Sort(m, l, h, hL, lG)
11     if (t ≤ hL)
12        then return Tth_Element(l, hL, t)
13     if (t ≥ lG)
14        then return Tth_Element(lG, h, t)
15     return A[t]
```

Рис. 2. Параметризованный алгоритм выбора

В алгоритме *The_Element* используются три вспомогательные процедуры: *Partition_Sort*, *MK*, *Insertion_Sort.*

Процедура *MK* предназначена для поиска медианы из $2k+1$ элементов, расположенных в массиве A с шагом $2q+1$ начиная с элемента с индексом $l+i$. Эта процедура своя для каждого *k*. Наилучшие верхние оценки количества операций сравнения в зависимости от *k* приведены в [4].

Insertion_Sort – стандартная процедура сортировки вставками, описанная, например, в [5].

Реализация алгоритма Дейкстры для решения задачи о голландском флаге [7], применительно к частичной сортировки приведена на *рис. 4.*

Аргументами процедуры *Partition_Sort* является величина m, относительной которой происходит частичная сортировка элементов глобального массива A в диапазоне $\{i \mid l \le i \;\&\; i \le h\}$. На рис. 3.1 представлен процесс формирования диапазонов L, E, G в цикле (*см. строки 2-12 на рис. 4*). Здесь диапазоны, представленные на *рис 3.1*, определены следующими соотношениями:

$$L = \{i \mid l \le i \;\&\; i < lX\}$$

$$X = \{i \mid lX \le i \;\&\; i < hX\}$$

$$E = \{i \mid hX < i \;\&\; i \le hE\}$$

$$G = \{i \mid hE < i \;\&\; hE \le h\}$$

В начале работы алгоритма диапазон X должен совпадать с исходным диапазоном, а остальные диапазоны должны быть пустыми. Это достигается операторами присваивания в строках 1 и 2 на *рис. 4.*

На *рис. 3.2* представлены отношения между переменными hL, lX и lG, hE и диапазонами L, E, G по завершению алгоритма. Сам алгоритм описан на *рис. 4.*

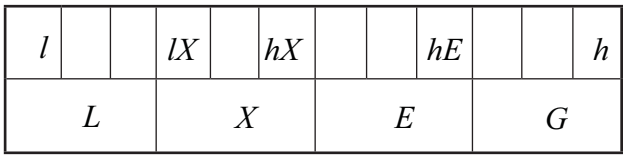

Рис. 3.1. Процесс формирования диапазонов L, E, G

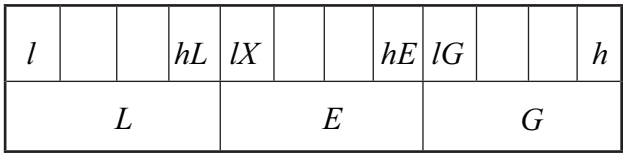

Рис. 3.2. Массив элементов после завершения работы процедуры *Partition_Sort*

Процедура *Partition_Sort* представлена на *рис. 4.*

```
Partition_Sort(m, l, h, hL, lG)
1  lX = l
2  hE = hX = h
3  while(lX ≤ hX)
4     r = A[hX]
5     if (r < m) then
6        A[hX] = A[lX]
7        A[lX++] = r
8     else if (r = m) then
9        hX--;
10    else // r < m
11       A[hX--] = A[hE]
12       A[hE--] = r
13 hL = hX
14 lG = hE + 1
```

Рис. 4. Алгоритм частичной сортировки

Обозначим через $T_k(n)$ – семейство оценок количества операций сравнения в худшем случае в алгоритме выбора зависящее от параметра *k*.

Поставим задачу исследовать как количество операций сравнения в алгоритме выбора, выполняемых в худшем случае, зависит от количества $2k+1$ элементов в группах (или параметра *k*). Для этого программно реализуем описанный алгоритм с параметризированной величиной $2k+1$ количества элементов в группе. На основе программной реализации алгоритма получим экспериментальные данные, содержащие количество сравнений в среднем случае и подтверждающие или опровергающие полученные теоретические результаты.

Оценка сложности алгоритма выбора в худшем случае

Оценим количество сравнений, требуемое для корректного завершения работы алгоритма выбора *t*-го элемента в порядке убывания из *n* элементов в худшем случае.

Обозначим через $T_k(n)$ количество сравнений, требуемое в худшем случае для корректного завершения работы алгоритма, в зависимости от параметра *k*. Анализируя описанный выше алгоритм (*рис. 2*) можно записать следующее рекуррентное уравнение:

$$\begin{cases} T_k(n) = T_k(2q+1) + T_k(n-(q+1)\cdot(k+1)+1) + \\ +(n-1) + V_k(2q+1), \ \textit{если } n > c \\ T_k(n) \le const, \ \textit{если } n < c \end{cases} \quad (2)$$

Здесь $q = (n/(2k+1)-1)/2$.

За вторым уравнением скрывается алгоритм сортировки вставками, поскольку этот алгоритм имеет наименьшую мультипликативную постоянную в оценке среди алгоритмов сортировки с квадратичной оценкой сложности [3]. Другими словами сортировка вставками является наиболее эффективной сортировкой на небольших размерах входного массива.

Разберем первое уравнение системы уравнений (2). Для нахождения медиан в $(2q+1)$ группах элементов, на которые разбивается входной массив, длинной $2k+1$ каждая, требуется $V_k \cdot (2q+1)$ сравнений, где V_k – наилучшая из известных верхних оценок поиска медианы в массиве из $(2k+1)$ элементов. Слагаемое $T_k(2q+1)$ соответствует числу сравнений, необходимых для нахождения медианы медиан (*рис. 2, строка 6*), закрашенных на *рис. 1* темным цветом. Число сравнений для частичной сортировки входного массива относительно найденной медианы равно $n-1$. Слагаемое $T_k(n-(q+1)\cdot(k+1)+1)$ определяет число сравнений в худшем случае (строки 8 – 11 листинга 1), необходимое для поиска *t*-го минимума в массиве, полученном после частичной сортировки входного массива относительной найденной медианы и отбрасывания элементов.

Существует большое число методов решения рекуррентных соотношений, многие из которых описаны в [8] и в [5]. Для решения соотношений (2) используем метод [9]. Поскольку $T_k(n)$ – линейная функция, будем искать ее в виде $c_1 n + c_2$.

В [3] доказано, что алгоритма выбора *t*-го элемента в порядке убывания из *n* является линейным. Соответственно количество операций сравнения, требуемых для нахождения *t*-го минимума, можно записать как

$$T_k(n) = c_1 n + c_2 = c_1 \cdot (2q+1)\cdot(2k+1) + c_2 .$$

Левые части данного уравнения и первого уравнения из системы (2) равны, а соответственно равны и правые части. Заменяя n на $(2q+1)\cdot(2k+1)$, получим следующее уравнение

$$c_1 \cdot (2q+1)\cdot(2k+1) + c_2 = c_1 \cdot (2q+1) +$$
$$c_2 + c_1\left[(2q+1)\cdot(2k+1) - (q+1)\cdot(k+1)+1\right]$$
$$c_2 + \left[(2q+1)\cdot(2k+1)-1\right] + V_k \cdot (2q+1)$$

Для того, чтобы данное равенство выполнялось при $n \to \infty$ (а соответственно при $q \to \infty$ поскольку связь линейная), необходимо выполнение следующих равенств:

$$\begin{cases} c_1 q - c_1 qk + 4kq + 2q + V_k 2q = 0, \\ c_1 + c_2 - c_1 k + 2k + V_k = 0 \end{cases} \quad (3)$$

В первое уравнение отнесем те члены уравнения, которые содержат q, а во второе уравнение вынесем константы. Данное действие можно проделать на основании того факта, что разные части уравнения растут по-разному. Решим систему линейных уравнений (3) относительно неизвестных констант c_1 и c_2. В итоге получим

$$\begin{cases} c_1 = \dfrac{2V_k + 4k + 2}{k-1}, \\ c_2 = V_k + 2k + 2 \end{cases} \quad (4)$$

Теперь минимизируем мультипликативную постоянную c_1 по k. Поскольку никакого общего метода нахождения медианы за наименьшее число сравнений даже небольших массивов чисел до сих пор не существует [4], воспользуемся *табл. 1*, данные которой собраны из нескольких источников – [4] и [10]. Отметим также, что в данную таблицу не входит разбиение массива на группы с четным числом элементов, поскольку подобное разбиение приведет к увеличению времени работы алгоритма.

Таблица 1.

Наилучшие из известных верхних оценок V_k выбора медианы из $2k+1$ элементов

k	2	3	4	5
$2k+1$	5	7	9	11
V_k	6	10	14	20

Подставляя значения V_k из *табл. 1* в систему уравнений (4), найдем оптимальное количество элементов в группе, на которое необходимо разбивать входной массив алгоритма поиска *t*-го элемента в порядке убывания. Результаты отображены в *табл. 2.*

Таблица 2.

Результаты подстановки значений V_k в (4)

k	$c_1(k)$	$c_2(k)$
2	22	12
3	17	18
4	15,(3)	24
5	15,5	32

Согласно полученным данным, разбиение входного массива на группы по семь элементов (при $k=3$) не является оптимальным. Верхняя оценка количества сравнений в данном случае $(17n+18)$ будет асимптотически хуже, чем верхняя оценка $(15,(3)n+24)$ для разбиения массива на группы с количеством элементов девять (при $k=4$). Таким образом, разбиение входного массива на группы по девять элементов будет лучшим.

Тем не менее, остается открытым вопрос: какова сложность алгоритма выбора при $k=5$. В соответствии с формулой (1) получим

$$V_6(11) \le 5 + 5 \cdot \lceil \log_2(7) \rceil = 20,$$

т.е. медиана может быть найдена не более чем за 20 сравнений. С другой стороны, для того чтобы при $k=5$ алгоритм имел лучшую мультипликативную постоянную необходимо, чтобы выполнялось неравенство

$$15,(3) < \frac{2V_5 + 4 \cdot 5 + 2}{5-1},$$

откуда $V_5 < 19\frac{2}{3}$.

Пока алгоритм, который позволяет найти медиану массива из 11 элементов за 19 сравнений не найден. Это открытая проблема.

До настоящего времени в литературе (например, [4], [5] и [6]) входной массив разбивался либо на 5, либо на 7 элементов. Это приводило к упрощению алгоритма, но не являлось оптимальным, как было показано выше.

Численный эксперимент

Для получения практических результатов, было проведено экспериментальное исследование программной реализации алгоритма, описанного в листинге 1, для *среднего случая*. Эксперимент проводился в диапазоне значений

$$k = \{5,7,9\};\ n = 5000 + i \cdot 1000,\ \textit{где}\ i = \overline{0,206}$$

Для каждой пары k и n проводилось 1000 экспериментов с генерацией различных значений элементов входного массива. Алгоритм генерации различных значений элементов входного массива представлен на *рис. 5*. В результате выполнения функции *Randomize* получаются случайные перестановки с равномерным распределением. Данное утверждение доказано в (Кормен, и др., 2007).

```
Randomize(A)
1  n ← length[A]
2  for i ← 1 to n
3     do Swap(A[i] ↔ A[Random(i, n)])
```

Рис. 5. Алгоритм генерации перестановок

В исходном коде программы, реализующей алгоритм выбора, были расставлены счетчики согласно методике, предложенной в [11], которые инкрементировались на единицу при очередном использовании операции сравнения. Помимо вычисления среднего количества операций сравнения, были также вычислены максимальное $(\max_n)$ и минимальное $(\min_n)$ число сравнений, размах числа сравнений $(\max_n - \min_n)$ и среднее квадратичное отклонение

$$\left(\hat{\sigma} = \sqrt{\sum_{i=0}^{n} (x_i - \bar{x}) \Big/ n-1} \right).$$

Отметим, что 1000 экспериментов с генерацией различных значений элементов входного массива является довольно малой величиной для точной оценки среднего квадратичного отклонения и размаха алгоритма. Тем не менее, на основе полученных данных, можно увидеть тренд роста данных показателей.

На *рис. 6* показана зависимость среднего числа сравнений от длины входного массива. На графике отчетливо видно, что разбиение входного массива на группы по девять элементов является оптимальным в *среднем случае*. Таким образом верность аналитических выкладок доказана экспериментальным путем.

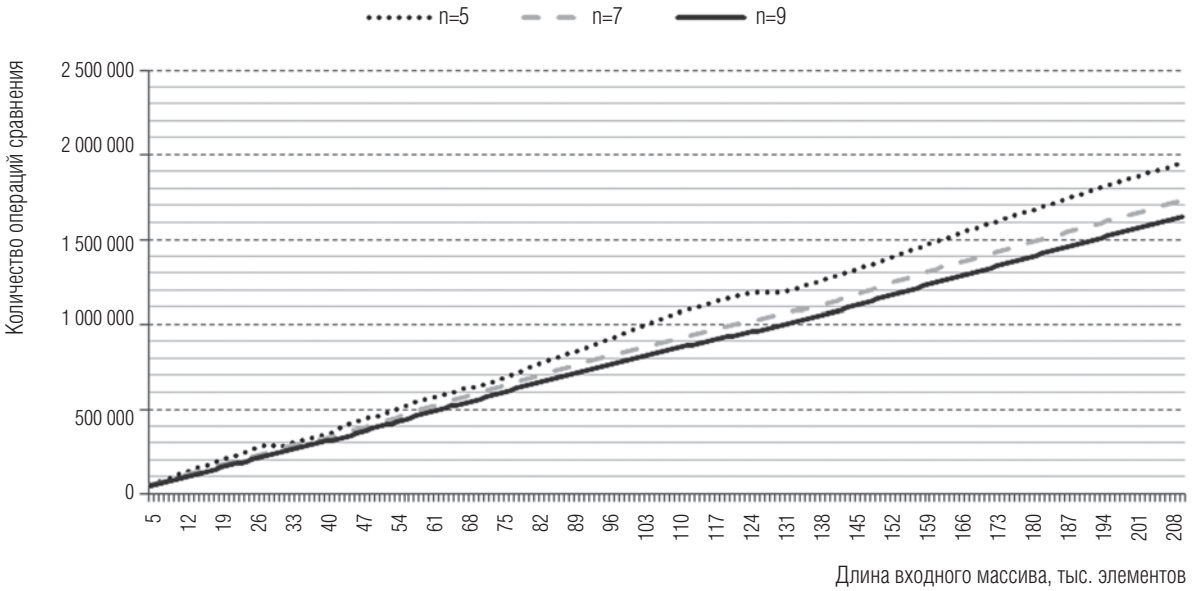

Рис. 6. Зависимость количества операций сравнения от длины входного массива

Для оценки среднего числа операций сравнения на основании данных, полученных в ходе описанного выше эксперимента, была построена однофакторная регрессия. Полученная формула регрессии имеет следующий вид: $\hat{T}(n) = 7,653642n + 14485.$ Характеристики построенной регрессии приведены в *табл. 3*. Отметим основные моменты: коэффициенты регрессии значимы, сама регрессия также значима, коэффициент детерминации стремится к 1, что говорит о том, что построенная модель близка к экспериментальным наблюдениям.

Таблица 3.

Характеристики построенной регрессии (представлено в формате EViews)

Variable	Coefficient	Std. Error	t-Statistic	Prob.
C	14485.24	1323.664	10.94329	0.0000
N	7.653642	0.010724	713.6855	0.0000

R-squared	0.999598	Mean dependent var	841078.6
Adjusted R-squared	0.999596	S.D. dependent var	458544.8
S.E. of regression	9219.791	Akaike info criterion	21.10571
Sum squared resid	1.74E+10	Schwarz criterion	21.13791
Log likelihood	-2182.441	Hannan-Quinn criter.	21.11873
F-statistic	509347.1	Durbin-Watson stat	0.015210
Prob(F-statistic)	0.000000		

Выводы

В статье предложена формализация задачи параметрической оптимизации мультипликативной постоянной в асимптотической оценке сложности алгоритма выбора и ее решение. В работе показано, что разбиение входного массива чисел на группы по семь элементов в алгоритме выбора *t*-го элемента является неоптимальным. В результате аналитического решения показано, что для алгоритма выбора (при известных алгоритмах поиска медианы при малых размерностях входных массивов данных) лучшим значением количества элементов в группе является девять. На основе программной реализации алгоритма получены экспериментальные данные, демонстрирующие верность теоретических выкладок. ■

Литература

1. Schreier J On tournament elimination systems [Журнал] // Mathesis Polska. - 1932 г.. - 7.
2. Blum M. [и др.] Time bounds for selection [Журнал] // J. Comput. System Sci.. - 1973 г.. - 7. - стр. 448-461.
3. Кнут Д. Э. Искусство программирования. Сортировка и поиск. [Книга]. - Москва : Вильямс, 2008. - Т. III.
4. Ахо А. В., Хопкрофт Д. Э. и Ульман Д. Д. Структуры данных и алгоритмы [Книга]. - Москва : Вильямс, 2007.
5. Кормен Т. [и др.] Алгоритмы. Построение и анализ [Книга]. - Москва : Вильямс, 2007.
6. Дейкстра Э. Дисциплина программирования [Книга]. - Москва : Мир, 1978.

7. Грэхем Р., Кнут Д. и Паташник О. Конкретная математика. Основание информатики [Книга]. - Москва : Мир, 2006.
8. Курант Рихард Курс дифференциального и интегрального исчисления [Книга]. - Москва :, 1967. - Т. 1 : 2.
9. Noshita K. Median Selection of 9 Elements in 14 Comparisons [Журнал] // Information Processing Letters. - 1974 г.. - 3(1). - стр. 8-12.
10. Hadian A. и Sobel M. Selecting the t-th largest using binary errorless comparisons. [Доклад] : Technical Report 121 / Dept. of Statistics ; University of Minnesota. - May, 1969.
11. Головешкин В. А. и Ульянов М. В. Теория рекурсии для программистов [Книга]. - Москва : Физматлит, 2006.